Laboratório 5 (parte 2) — Implementação de um serviço de nomeação simples baseado em tabela hash distribuída

Sistemas Distribuídos (MAB-367)
Profa. Silvana Rossetto

[1]DCC/IM/UFRJ

Introdução

O objetivo deste Laboratório é implementar uma versão simplificada do protocolo **Chord**: um exemplo de serviço de nomeação (e localização) baseado em **tabela hash distribuída**.

Artigo de referência sobre o Chord : *Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on networking, 11(1), 17-32.*

Descrição do problema[1]

Para limitar o escopo do problema, vamos implementar uma **tabela hash distribuída imutável**, baseada no Chord. Os nós serão organizados em um anel lógico com 2^n nós. Cada nó i saberá quem é seu **nó sucessor** e manterá uma **tabela de apontamentos** (*finger table*, como definida no protocolo Chord).

Vamos simular a execução da tabela hash distribuída em um única máquina. O programa principal receberá como entrada o valor de n, escolherá **portas distintas para cada nó** e disparará um **processo filho** para simular a execução de cada um dos nós, montando o anel lógico com identificadores consecutivos.

Inserção dos pares chave/valor quando um novo par chave/valor for inserido na tabela, ele deverá ser associado/armazenado no primeiro nó cujo identificador é maior ou igual ao valor *hash* gerado a partir da chave dada. A operação `insere(noOrigem, chave, valor)` deverá ser implementada, onde `noOrigem` é o nó para o qual a aplicação cliente está fazendo a solicitação. Para determinar qual nó irá armazenar o par, o `noOrigem` deverá gerar o *hash* da chave, determinar o seu nó de destino e depois rotear a mensagem de insersão até esse nó.

Busca por chaves qualquer nó da tabela poderá ser contactado pela aplicação cliente para localizar uma determinada chave. A operação `busca(idBusca, noOrigem, chave)` deverá ser implementada, onde `noOrigem` é o nó para o qual a aplicação cliente está fazendo a solicitação. O `noOrigem` deverá gerar o hash da chave para saber qual nó contém o valor correspondente e deve encaminhar a consulta para esse nó. Quando a consulta chegar ao seu destino, o nó deverá enviar o valor correspondente de volta para a aplicação cliente. Para isso, deve ser incluída uma referência para a aplicação cliente junto com a solicitação de consulta, para que o nó que armazena o par chave/valor possa enviar o resultado diretamente para o cliente, como ilustrado na figura1. O parâmetro `idBusca` servirá para diferenciar as operações de busca.

[1]Baseada na descrição do exercício de implementação 2 definido aqui: https://www.cs.rpi.edu/academics/courses/fall19/proglang/

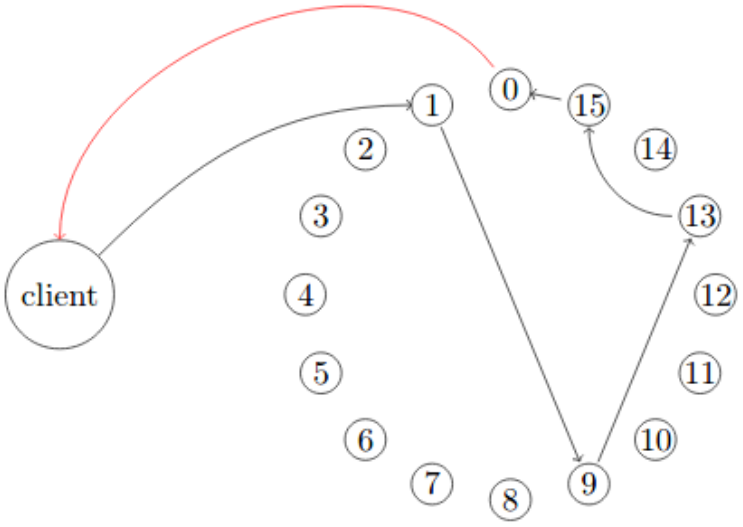

Figura 1. Retorno do valor para a aplicação cliente em uma operação de busca na tabela.

Simulação da interface de acesso aos nós Depois de disparados os processos filhos, o programa principal deverá ficar disponível para responder consultas sobre os nós da tabela e sua localização (IP e porta).

Aplicação cliente A aplicação cliente deverá permitir ao usuário realizar **inserções de pares chave/valor e consultas às chaves**, indicando sempre um nó de origem. Os resultados das consultas deverão exibir o valor da chave procurada e o nó onde ela foi encontrada. As inserções e consultas deverão ser encaminhadas diretamente para o nó de origem (sem passar pelo programa principal).

Implementação

A implementação deste trabalho poderá ser feita usando a API de sockets diretamente ou qualquer uma das abstrações que estudamos (RPC, fila de mensagens).

Entrega do laboratório Use o **formulário de entrega desse laboratório** para enviar o link do repositório do código implementado.